付録：
クラスルーム PC 管理ソフトウェア
修正箇所報告

はじめに

本ドキュメントは，Red Hat Enterprise Linxu ES4 Update2 を管理サーバーに用いた際に，CEC 版クラスルーム PC 管理ソフトウェア 1.3-3 のソースコードに行った変更，及び，マニュアル記載事項と異なった指定を行った箇所を記すものである．

なお，このドキュメントに記載されている変更は，以下の構成に対して行ったものである．

対象		内容
管理サーバー	ハードウェア 管理サーバー OS ｸﾗｲｱﾝﾄ管理 OS	「京田辺市/柏市」 FUJITSU PRIMERGY TX150 Red Hat Enterprise Linux ES release 4 (Nahant Update 2) KNOPPIX Linux 4.0.2 (knoppix_v4.0.2CD_20050923_xen3.0vt-20060126+IPAFont_cdsize.iso)
参照 PC クライアント PC	ハードウェア ＯＳ	「京田辺市」 富士通 FMV-686NU（メモリ合計 512MB もしくは 384MB） Turbolinux Desktop 11 (Fuji) Turbolinux HOME 「柏市」 マウスコンピュータ LUV BOOK i1000DW-N(メモリ合計 512MB) Turbolinux Desktop 11 (Fuji)

1. クラスルーム PC 管理ソフトウェアに関する注意事項

（1）ネットワーク接続上の注意点

クライアント PC の仕様によっては、無線 LAN を有効にする為のハードウェアスイッチをそなえている場合がある。システム起動時に、このスイッチが OFF になっているために無線 LAN に接続できない場合があるので注意が必要である。

（2）クライアント PC の故障時の注意点

クライアント PC が故障した際には、予備機を代替として利用する。但し、ハードウェア（特に NIC）が故障した場合は、修理完了後に Mac アドレスが変わっている可能性があるので注意が必要。Mac アドレスが変更になると、静的に割り当てている IP アドレス（DHCP 機能）が有効にならない。その場合は、サーバ側の設定ファイルを更新する必要がある。

（3）PC 管理ソフトウェアの導入環境について

PC 管理ソフトウェアを RedHatEnterpriseLinuxES4 で動作させる為に、spcman-1.3-3 のソースコードを修正している。以下に、修正内容を記す。

クライアント PC が、Serial ATA 経由でハードディスク接続を行っている場合、Python ソースコード内にハードディスクデバイス記述が「hda」とハードコードされているため、SATA 機では動作しないことが確認されている 。

ソースコード中の「hda」を「sda」と記述しなおすことで対応する。ただしこの場合、管理対象の参照 PC、クライアント PC に従来通りの IDE 接続ハードディスクなど「hda」で接続するものが混在していた場合、これらの PC に対するイメージ取得、およびイメージ配布を行うことはできない。

【修正前】（/opt/IBM/SPCMan/script/agent/LinuxCommon.py ）

```
37    self.MOUNT_DIR = "/mnt/"
38    self.NFS_MOUNT_DIR = self.MOUNT_DIR + "nfs/"
39    self.DEVICE_DIR = "/dev/"
40    self.HDD_DEVICE = self.DEVICE_DIR + "hda"
41    self.SWAPFS = "linux-swap"
```

【修正後】（40 行目　「hda」を「sda」に修正 ）

```
37    self.MOUNT_DIR = "/mnt/"
38    self.NFS_MOUNT_DIR = self.MOUNT_DIR + "nfs/"
39    self.DEVICE_DIR = "/dev/"
40    self.HDD_DEVICE = self.DEVICE_DIR + "sda"
41    self.SWAPFS = "linux-swap"
```

2. イメージ配布・展開後の自動電源 OFF

参照 PC からのイメージ，クライアント PC へのイメージ展開が終了したのち，本来自動で電源 OFF されるべき対象 PC の電源が落ちないことが確認されている．

『クラスルーム PC 管理ソフトウェアマニュアル-管理ソフトウェア導入手順』の p 11 4-②「pxelinux.cfg/default を編集します」に記載されている編集例に加えて，PXE 経由で起動させている KNONPPIX のカーネルパラメータに，noprompt，noeject を記載することで電源が落とすことができるようになる。

【修正前】　導入手順 p11　5.2-4 pxelinux.cfg/default を編集の②に記載されている例

```
2    APPEND secure nfsdir=192.168.0.2:/repository/nfs-root nodhcp lang=us
                ramdisk_size=100000 apm=power-off nomce vga=791
                initrd=miniroot.gz quiet BOOT_IMAGE=knoppix 1
```

【修正後】

カーネルオプション「noprompt」，「noeject」を追記

```
2    APPEND secure nfsdir=192.168.0.2:/repository/nfs-root nodhcp lang=us
                ramdisk_size=100000 apm=power-off nomce vga=791
                initrd=miniroot.gz quiet noprompt noeject BOOT_IMAGE=knoppix 1
```

3. 保護ディレクトリ機能の有効化

保護ディレクトリ機能を有効にした際，イメージ展開時に umount に失敗して展開が完了できない現象が確認されている．

【修正前】 MRI 版 /opt/IBM/SPCMan/script/agent/CustomCommand.py

```
51  def umount(self, dir, fstype):
52      """Unmounts the directory"""
53      # if this file system is a swap file system, doesn't unmount,
54      # just return
55      if fstype == self.LINUX_SWAP_TYPE:
56                      return
57
58      command = "umount -f " + dir
59      if self.exec_command(command) != 0:
60          DebugPrint.print_msg("Umount Error!!!")
61          raise CommandException, "unmount error: "+ command
```

【修正後】

58 行目の umount コマンドについて，「-l」オプションを追記

```
51  def umount(self, dir, fstype):
52      """Unmounts the directory"""
53      # if this file system is a swap file system, doesn't unmount,
54      # just return
55      if fstype == self.LINUX_SWAP_TYPE:
56                      return
57
58      command = "umount -f -l " + dir
59      if self.exec_command(command) != 0:
60          DebugPrint.print_msg("Umount Error!!!")
61          raise CommandException, "unmount error: "+ command
```

4. Redhat による WOL 経由の電源 ON

Wake On LAN で電源を ON にする際，管理サーバーから ether-wake コマンドを実行している．SUSE と Redhat では，このコマンドの使用法が異なるため，Python コード中に記載されている ether-wake コマンド発行法から修正を入れる必要がある．

Redhat では，-i オプションを付けないと起動させられないが，SUSE ではこのオプションが必要ない．

- Redhat： *ether-wake -i <インターフェイス名> MAC アドレス*
- SUSE ： *ether-wake MAC アドレス*

【修正前】spcman-1.3-3 /opt/IBM/SPCMan/script/schedule/Monitor.py

```
45  def kick(self, term_list):
46      for term in term_list:
47          term = self._refresh(term)
48          if term.get_status()==str(Status.DEPLOY_WAITING) or term.get_status()==str(Status.FETCH_WAITING):
49              startup_pc_num = self.checkStartupPC()
50              if startup_pc_num < self.STARTUP_PC_NUM:
51                  self.update_wakeup_time_mac(time.time(),term.get_mac_addr())
52                  if term.get_status()==str(Status.FETCH_WAITING):
53                      self.status_manager.update_status(Status.FETCH_STARTING, term.get_mac_addr())
54                  elif term.get_status()==str(Status.DEPLOY_WAITING):
55                      self.status_manager.update_status(Status.DEPLOY_STARTING, term.get_mac_addr())
56
57                  if os.system("../../bin/ether-wake" + " " + term.get_mac_addr()) != 0:
58                      DebugPrint.print_msg("wakeup error!")
59              time.sleep(10)
```

【修正後】

57 行目　ether-wake コマンドに「-i eth0」オプションを追記

```
45  def kick(self, term_list):
46      for term in term_list:
47          term = self._refresh(term)
48          if term.get_status()==str(Status.DEPLOY_WAITING) or term.get_status()==str(Status.FETCH_WAITING):
49              startup_pc_num = self.checkStartupPC()
50              if startup_pc_num < self.STARTUP_PC_NUM:
51                  self.update_wakeup_time_mac(time.time(),term.get_mac_addr())
52                  if term.get_status()==str(Status.FETCH_WAITING):
53                      self.status_manager.update_status(Status.FETCH_STARTING, term.get_mac_addr())
54                  elif term.get_status()==str(Status.DEPLOY_WAITING):
55                      self.status_manager.update_status(Status.DEPLOY_STARTING, term.get_mac_addr())
56
57                  if os.system("../../bin/ether-wake -i eth0" + " " + term.get_mac_addr()) != 0:
58                      DebugPrint.print_msg("wakeup error!")
59              time.sleep(10)
```

また，上記 ether-wake コマンドファイルは，インストール時に，/usr/sbin/ether-wake からコピーして /opt/IBM/SPCMan/bin/以下に配置する設定になっている(spcman.spec)．しかし，Redhat の ether-wake コマンドは，「/sbin/ether-wake」に配置されているため，インストール時のコピーに失敗し，電源 ON ができない．

このため，redhat では，/sbin/ether-wake をコピー元に使用するよう spcman.spec を書き換えたのち，rpm を再生成する。

【修正前】 spcman.spec

```
76  if [ -x /usr/sbin/ether-wake ] ; then
77  cp -f /usr/sbin/ether-wake /opt/IBM/SPCMan/bin/ether-wake
78  else
79  echo "echo no ether-wake command found" > /opt/IBM/SPCMan/bin/ether-wake
80  chmod a+x /opt/IBM/SPCMan/bin/ether-wake
81  fi
82  chmod u+s /opt/IBM/SPCMan/bin/ether-wake
```

【修正後】 spcman.spec

```
76  if [ -x /sbin/ether-wake ] ; then
77  cp -f /sbin/ether-wake /opt/IBM/SPCMan/bin/ether-wake
78  else
79  echo "echo no ether-wake command found" > /opt/IBM/SPCMan/bin/ether-wake
80  chmod a+x /opt/IBM/SPCMan/bin/ether-wake
81  fi
82  chmod u+s /opt/IBM/SPCMan/bin/ether-wake
```

5. CustomCommand.py 修正

CustomCommand.py について、修正を行う。

5-1. Python インスタンス変数への参照記法修正

Python のクラス内で宣言した変数を参照する場合，「self.xxxx」という記法を行う必要がある．コード内にグローバル変数への参照方法で記述されていた箇所を，インスタンス変数参照記法に変更する。

【修正前】

```
220   command = CHROOT + root_dir + " " + command
```

【修正後】

220 行目　CHROOT への参照時に，「self.」を追記

```
220   command = self.CHROOT + root_dir + " " + command
```

※同様の修正を，以下の箇所に加える。(数字は行数)

CHROOT　=> self.CHROOT　164, 187, 198
MOUNT_ALL => self.MOUNT_ALL　187, 198, 220

5-2. mount コマンドのパス修正

OS の mount コマンドを発行する箇所で，mount コマンドへのパスが「/sbin/mount」と定義されていたが，Redhat では，「/bin/mount」に存在するため，この箇所を修正した．

【修正前】

```
22   class CustomLinuxCommand(LinuxCommandList):
23       LINUX_SWAP_TYPE = "linux-swap"
24       REISER_FS = "reiserfs"
25       CHROOT = " /usr/sbin/chroot "
26       MOUNT_ALL = " /bin/rm /etc/mtab; /sbin/mount -a -t noproc,devpts,nfs,nfs4,smbfs "
```

【修正後】

26 行目　「/sbin/mount」コマンドのパスを「/bin/mount」に修正

```
22   class CustomLinuxCommand(LinuxCommandList):
23       LINUX_SWAP_TYPE = "linux-swap"
24       REISER_FS = "reiserfs"
25       CHROOT = " /usr/sbin/chroot "
26       MOUNT_ALL = " /bin/rm /etc/mtab; /bin/mount -a -t noproc,devpts,nfs,nfs4,smbfs "
```

6. Redhat におけるスケジュール起動対応

クラスルーム PC 管理システムにおけるスケジュール機能は， OS の cron デーモンを使用して実現されており，予め設定された時間に「/opt/IBM/SPCMan/script/fetch_deploy.sh」を実行することで起動される．Redhat で cron を実行する際は，環境変数が無効化された状態で実行される．そのため，実行ユーザの環境変数であったとしても，cron 実行前に再設定する必要がある．

今回は，環境変数を設定したのちにスケジュール機能(Python の Main)を起動する redhat_fetch_deploy.sh というファイルをクーロンから起動させる。

6-1. 環境変数の再設定

以下の内容を記述した/opt/IBM/SPCMan/script/redhat_fetch_deploy.sh を新規作成.

```
#!/bin/sh -f

export SPCMANROOT=/opt/IBM/SPCMan
export SPCMANSCRIPT=$SPCMANROOT/script
export SPCMANDBPATH=$SPCMANROOT/database/
export SPCMANEVENTLOGPATH=$SPCMANROOT/eventlog/

export PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin

export PYTHONHOME=/usr/lib/python2.3
export PYTHONPATH=$PYTHONHOME:$PYTHONHOME/site-packages:$PYTHONHOME/site-packages/gtk-2.0:$PYTHONHOME/site-packages/gtk-2.0/gtk:$PYTHONHOME/encodings:$PYTHONHOME/plat-linux2:$PYTHONHOME/lib-dynload:$SPCMANSCRIPT:$SPCMANSCRIPT/database:$SPCMANSCRIPT/common:$SPCMANSCRIPT/schedule

#export DISPLAY=127.0.0.1:0.0
#/usr/X11R6/bin/xhost +

cd $SPCMANSCRIPT/schedule; /usr/bin/python ./Main.py
```

6-2. cron から起動されるシェルスクリプトの変更

Python プログラムでは，スケジュール設定が保存されたタイミングで cron 設定ファイルを生成し，そのファイル内に起動スクリプトファイル(fetch_deploy.sh)を指定している．今回は，上記で示した独自のファイル（redhat_fetch_deploy.sh)を使用するため，以下の箇所を修正する。

【修正前】 spcman-1.3-3 /opt/IBM/SPCMan/script/console/spcman_p10.py

```
157     '''Write a file for cron'''
158     schedulecron_filepath = __schedulepath+"/Schedule/schedulecron"
159     schedulecron_file = __builtin__.open(schedulecron_filepath,'w')
160     cmd = "%s %s %s %s %s %s/fetch_deploy.sh \n" % (minutes ,hours ,
161                             day, month,day_of_week,
162                                     __spcmanscriptpath)
```

【修正後】

160 行目　fetch_deploy.sh を，redhat_fetch_deploy.sh に変更

```
157     '''Write a file for cron'''
158     schedulecron_filepath = __schedulepath+"/Schedule/schedulecron"
159     schedulecron_file = __builtin__.open(schedulecron_filepath,'w')
160     cmd = "%s %s %s %s %s %s/redhat_fetch_deploy.sh \n" % (minutes ,hours ,
161                             day, month,day_of_week,
162                                     __spcmanscriptpath)
```

7. 既存 PC への対応（京田辺市プロジェクトのみ適用）

今回､児童・生徒機として使用している FMV-686NU 向けに，クラスルーム PC 管理ソフトウェアを導入する際に必要な対応は以下の通りである．

7-1. ディスクジオメトリ情報の追加

FMV-686NU は BIOS でヘッダ数を 63 以上の値にしない。このため、ディスク容量を Knoppix 起動時に正確につたえられないため、起動時パラメータで設定しておく必要がある。既存環境では，パラメータ指定がないため、/tftpboot/X86PC/LINUX/pxelinux.cfg/default ファイルの以下の箇所を修正する必要がある．

```
2    APPEND secure nfsdir=192.168.0.2:/repository/nfs-root nodhcp lang=us
                ramdisk_size=100000 apm=power-off nomce vga=791
                initrd=miniroot.gz quiet hda=1835,255,63 BOOT_IMAGE=knoppix 1
```

7-2. イメージ保管ディレクトリの変更

参照 PC から取得した OS イメージは，tar.gz 形式ファイルで，/opt/IBM/SPCMan/image 以下に参照 PC ごとに分けて保管される．

今回の構成の場合，1 イメージあたりのファイルサイズは，約 2GB となり，10 世代管理する仕様から算出すると，約 20GB のディスク容量を確保する必要がある．今回，/opt が存在するマウントポイント「/」のパーティションサイズは 5GB しか確保されていないため，以下の，回避策を講じる．

7-2-1. イメージ保管ディレクトリの作成

今回の構成では，100GB とパーティションサイズに余裕のある「/home」にイメージを保管することとする．/home 以下にイメージ保管用ディレクトリを作成する．

```
[root@server root]# mkdir /home/spcman_image
[root@server root]# chmod 777 /home/spcman_image
[root@server root]#
```

7-2-2. シンボリックリンクの作成

プログラムでは，/opt/IBM/SPCMan/script がイメージ保存ディレクトリに固定されているため，従来の/opt/IBM/SPCMan/image を，先ほど作成した/home/spcman_image に対するシンボリックリンクとすることで対応する．

```
[root@server root]# mkdir  /home/spcman_image
[root@server root]# rm –fr /opt/IBM/SPCMan/image
[root@server root]# ln –s /home/spcman_image  /opt/IBM/SPCMan/image
[root@server root]# chmod 777 /home/spcman_image
[root@server root]#
```

7-2-3. /etc/exports の編集

イメージ保管ディレクトリは，イメージ配布時・イメージ展開時にクライアント PC 上の KNOPPIX から NFS マウントされて参照される．この際，NFS で公開するためには，これまでの「/optIBM/SPCMan」に加えて，「/home/spcman_image」も公開する必要がある．

```
/home    * (rw, no_root_squash)
/opt/IBM/SPCMan          192.168.1.0/255.255.255.0(rw,no_root_squash)
/repository/nfs-root/usr     192.168.1.0/255.255.255.0(rw,no_root_squash)
/repository/nfs-root 192.168.1.0/255.255.255.0(rw,no_root_squash)
/home/spcman_image 192.168.1.0/255.255.255.0(rw,no_root_squash)
```

7-2-4. configs.tbz の入替

configs.tbz には，/opt/IBM/SPCMan/image がシンボリックリンクになったことに伴って，シンボリックリンク先の参照ディレクトリ(/home/spcman_image)の実体と，そのディレクトリに対して，サーバー側のイメージ保管ディレクトリ(192.168.1.100:/home/spcman_image)を NFS 経由で mount する設定を，/etc/fstab に記述する．

```
[root@server root]# tar xvfz /opt/IBM/SPCMan/script/server-settigs.tar.gz
    (解凍処理結果)
[root@server root]# cd settings
[root@server root]# tar xvfj configs.tbz
    (解凍処理結果)
[root@server root]# mkdir home
[root@server root]# mkdir home/spcman_image
[root@server root]# vi /etc/fstab
```

vi /etc/fstab の編集内容

```
/proc     /proc       proc   defaults          0 0
/sys      /sys        sysfs  noauto            0 0
/dev/pts  /dev/pts    devpts mode=0622         0 0
/dev/fd0  /mnt/auto/floppy auto   user,noauto,exec,umask=000    0 0
/dev/cdrom /mnt/auto/cdrom  auto   user,noauto,exec,ro 0 0
192.168.1.100:/opt/IBM/SPCMan     /mnt/nfs        nfs     rw,nolock,tcp     1 1
192.168.1.100:/home/spcman_image    /home/spcman_image      nfs     rw,nolock,tcp   1 1
```

※ NFS マウントしてサーバー側のディレクトリ内にあるシンボリックリンクを参照した場合，リンク先は，ローカルディスク(クライアントマシン側)上のディレクトリを指し示している．

7-3. Turbolinux での wake-on-LAN 有効化

インストール状態では、eth0 デバイスは BIOS で Wake-on-LAN 設定をしても、起動しない。
Wake On LAN（WOL)は BIOS 設定だけの問題ではなく、OS の LAN ドライバが，BIOS 情報（ACPI)設定をするからである。
Windows の場合は LAN ドライバのプロパティで設定できるが，Linux の場合は ethtool コマンドを使用する。
ethtool コマンドは Turbolinux FUJI では含まれている。

#/usr/sbin/ethtool eth0

を実行し，表示に

wol d

が含まれていたら，WOL を禁止するように LAN ドライバが BIOS に指示している。
そこで、

#/usr/sbin/ethtool -s eth0 wol g

で WOL を MagicPaket で起動する設定にして，再度、

#/usr/sbin/ethtool eth0

で「wol g」になっていることを確認し，この状態で，shutdown する。
これで PC 管理システムの参照 PC からイメージ採取を実行すると 10 秒後くらいに電源が入る。

#/usr/sbin/ethtool eth0

で「wol g」にならないのであれば，LAN ドライバが古いので，LAN ドライバの入れ替えが必要である。

また，Turbolinux 10D や Turbolinux HOME ではインストールされないため，

http://sourceforge.net/projects/gkernel/

からダウンロードし，コンパイルする必要がある。rpm も入手できるが古いので，ソースから./configure し、make;make install する。

今回使用した Turbolinux HOME では、/usr/local/sbin/ethtool で実行できるようインストールした。